Cubed: Bounded-Memory Serverless Array Processing (in Xarray)

*Tom Nicholas
Tom White

Worthy
GORDON AND BETTY MOORE FOUNDATION
Lamont-Doherty Earth Observatory
COLUMBIA UNIVERSITY | EARTH INSTITUTE

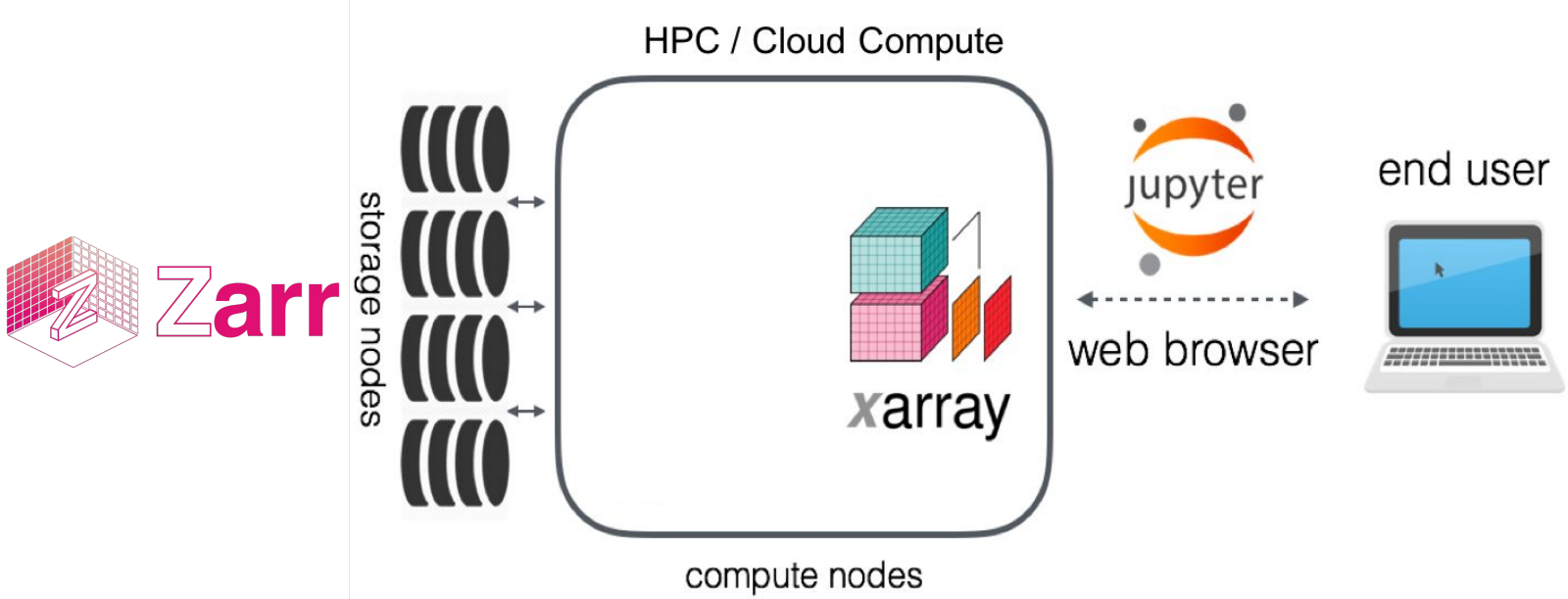
*tom@cworthy.org
@TomNicholas

What I will talk about:

- Vision for Science at Scale
- What is Cubed?
- Xarray integration
- Initial results
- Pros and Cons
- Next steps

Vision for Science at Scale (Tom's list)

- My perfect parallel executor…



Vision for Science at Scale (Tom's  list ) - (1)

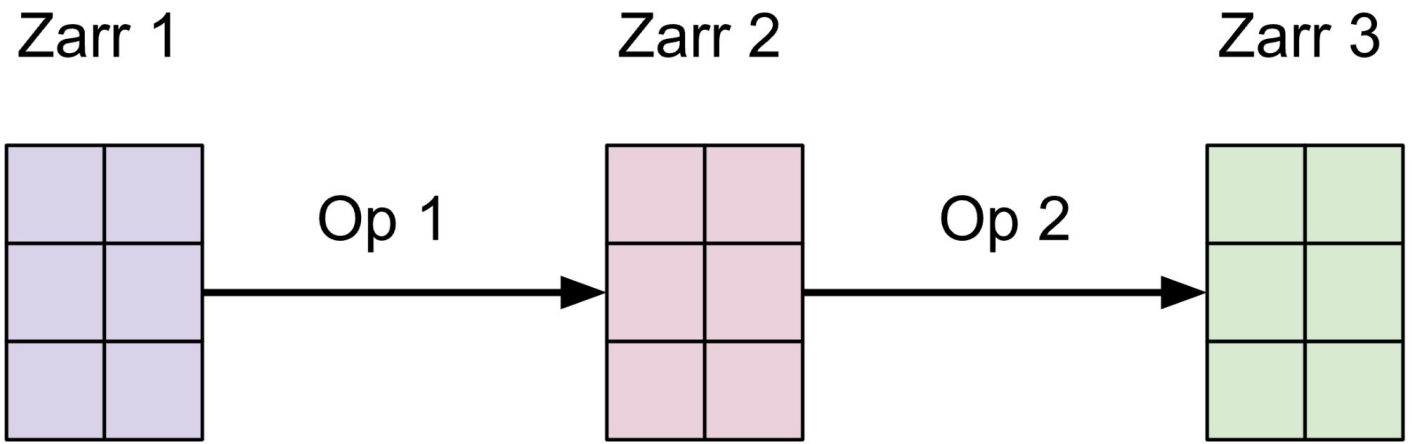
- Expressive
  - Scale without rewriting
- Perfect weak horizontal scaling
  - (1000x problem in 1x time with 1000x CPUs)
- Predictable (no nasty RAM surprises)
- Forget about the Cluster

Vision for Science at Scale (Tom's  list ) - (2)

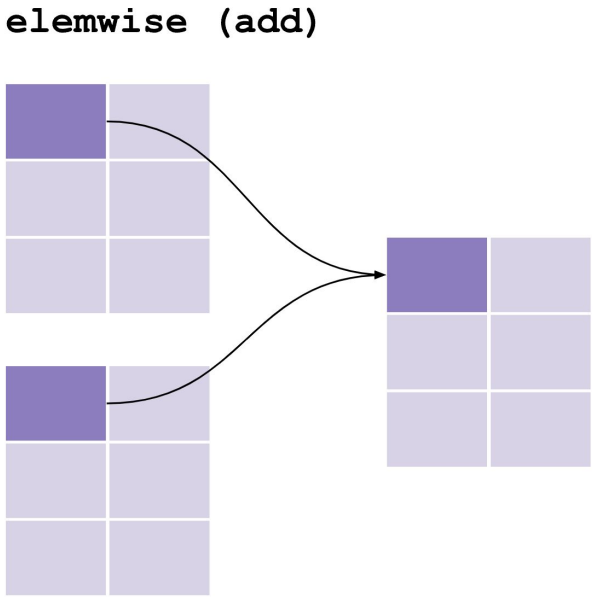
- …
- Robust to small failures
- Resumable
- Fully open
  - Not locked in to any one service, platform, or knowledge base

What is Cubed?

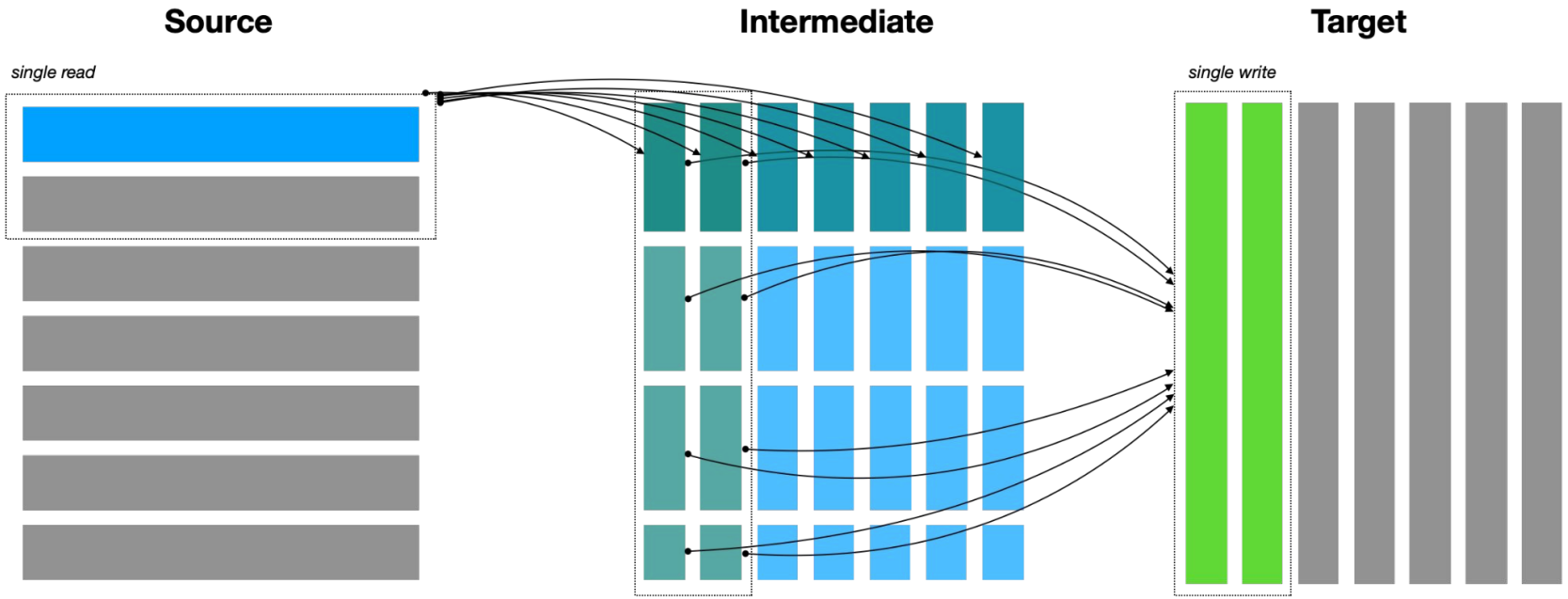
- Idea: All array operations take Zarr -> Zarr



- Many simple array operations are “blockwise”

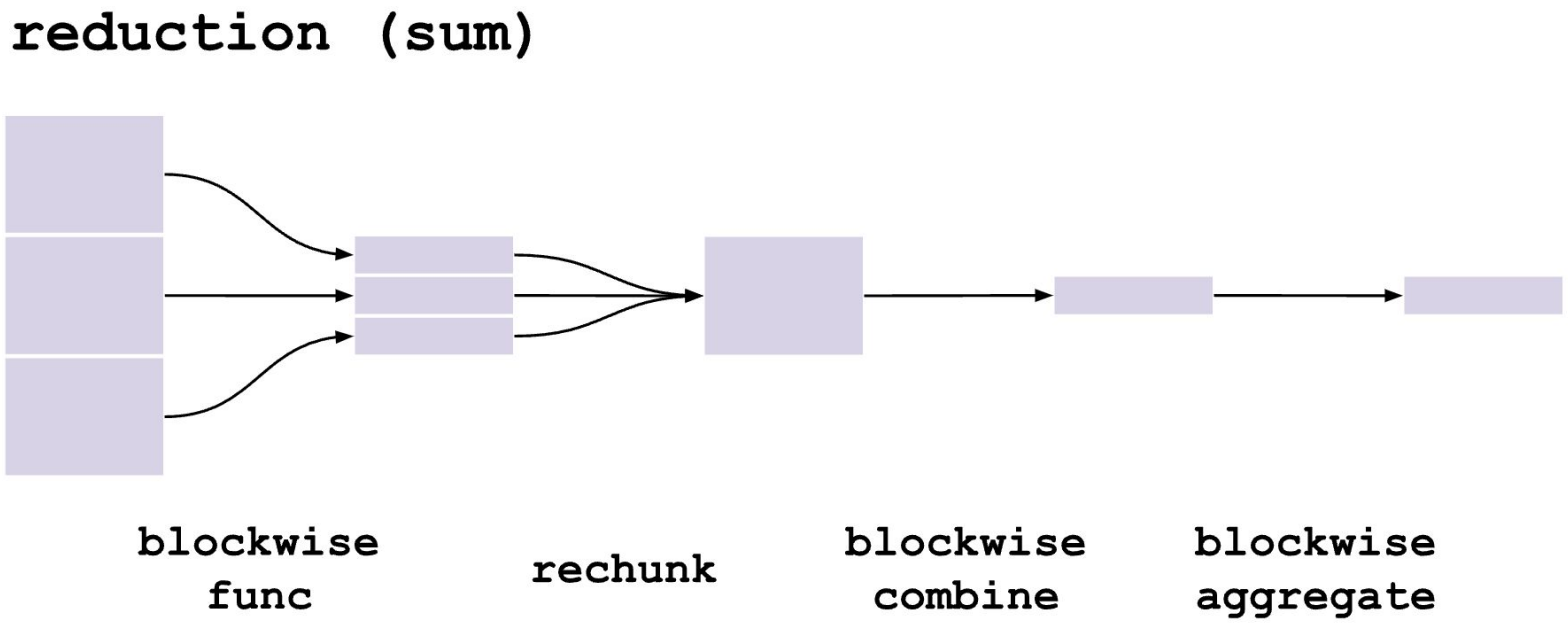


- But some require a “rechunk”

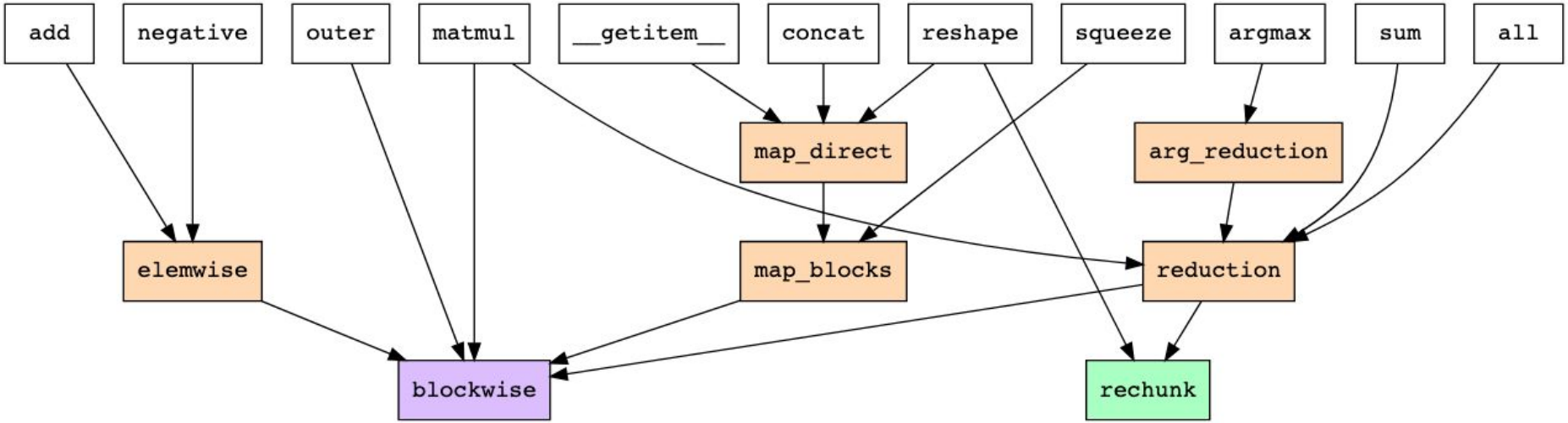


(diagram from pangeo-data/rechunker package)

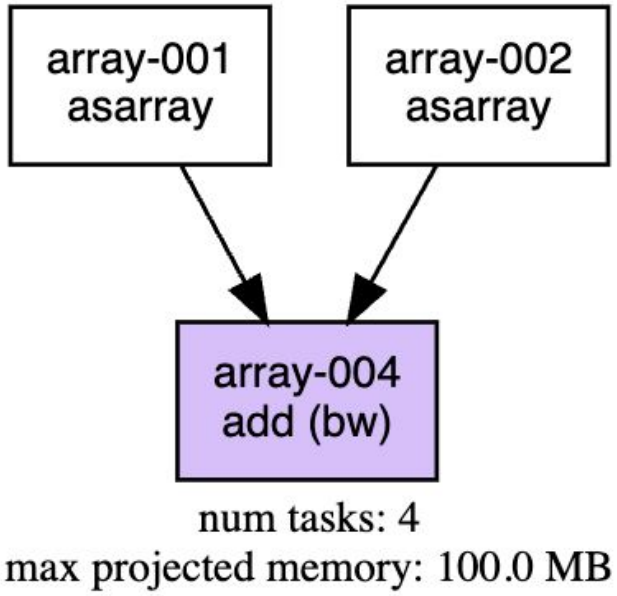
- Some operations require both “blockwise” and “rechunk”



- Turns out blockwise + rechunk cover all numpy-like array operations!



- Chain operations into a high-level “Plan”
  - Represented at array-level, not chunk-level



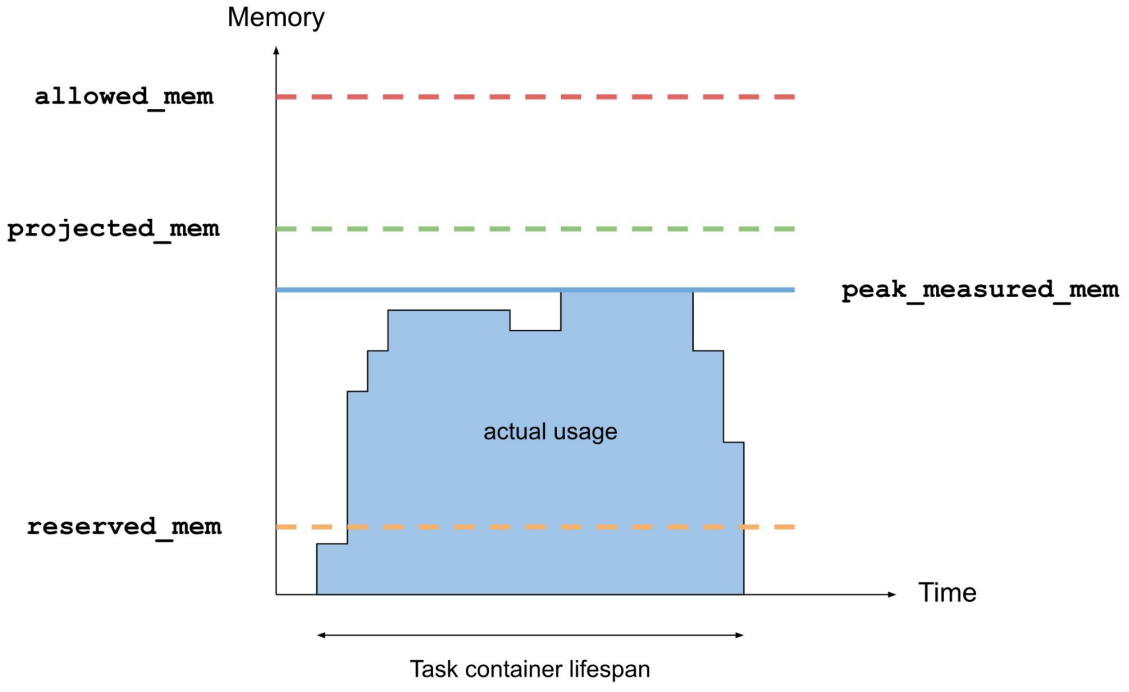
Bounded-memory operations

- Blockwise processes one chunk at a time

- Rechunk can be constant memory if intermediate Zarr store
  - (see pangeo Rechunker package)

Bounded-memory operations

- Can therefore predict memory usage before launching!

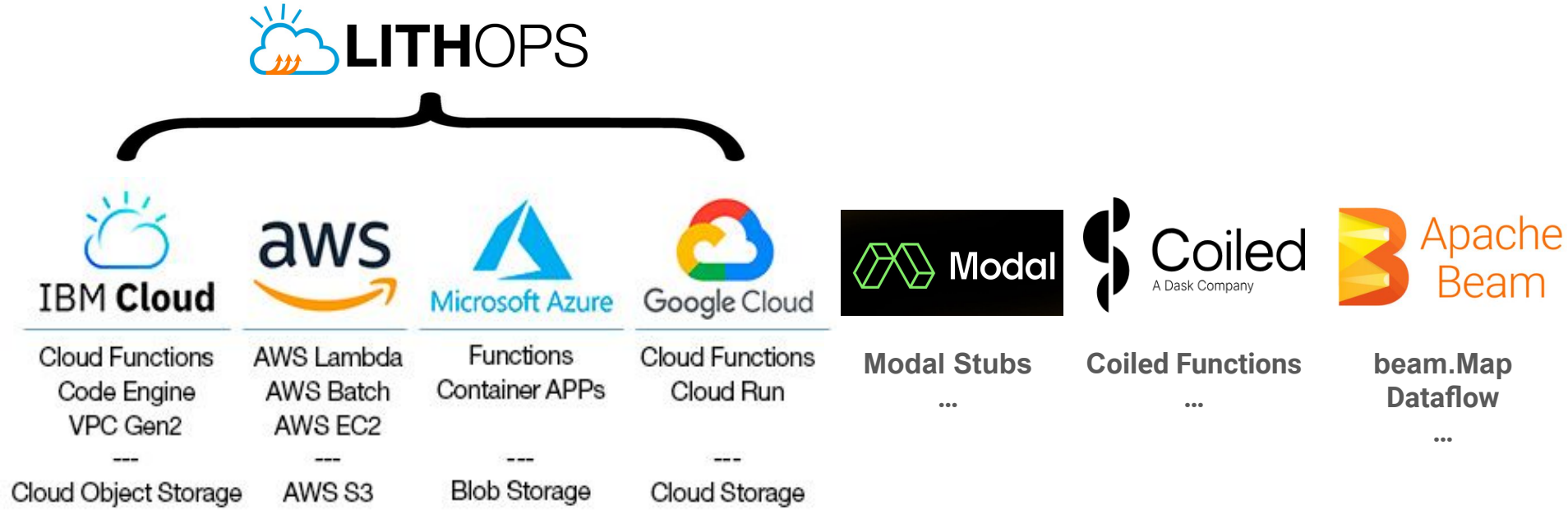


Serverless execution

- Every op is (a series of) embarrassingly parallel tasks
- Just launch them all simultaneously
- Ideal fit for ✨Serverless✨ cloud services
  - e.g. AWS Lambda, Google Cloud Functions
- (Means no Cluster to manage!)

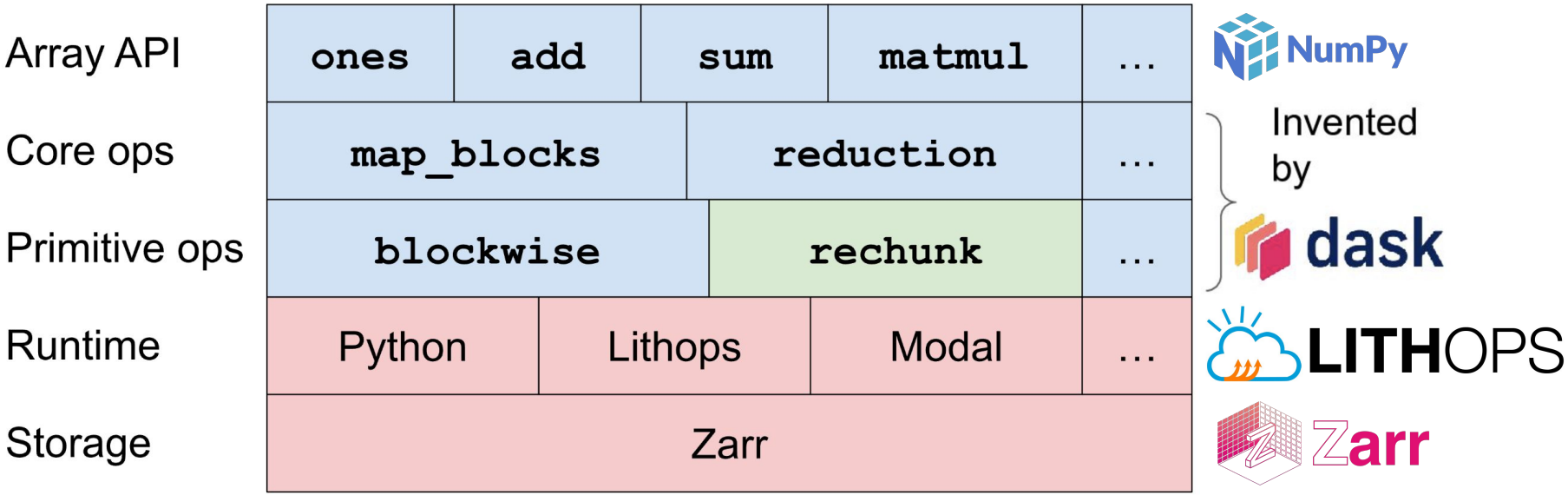
Range of Executors

- Abstract over cloud vendors



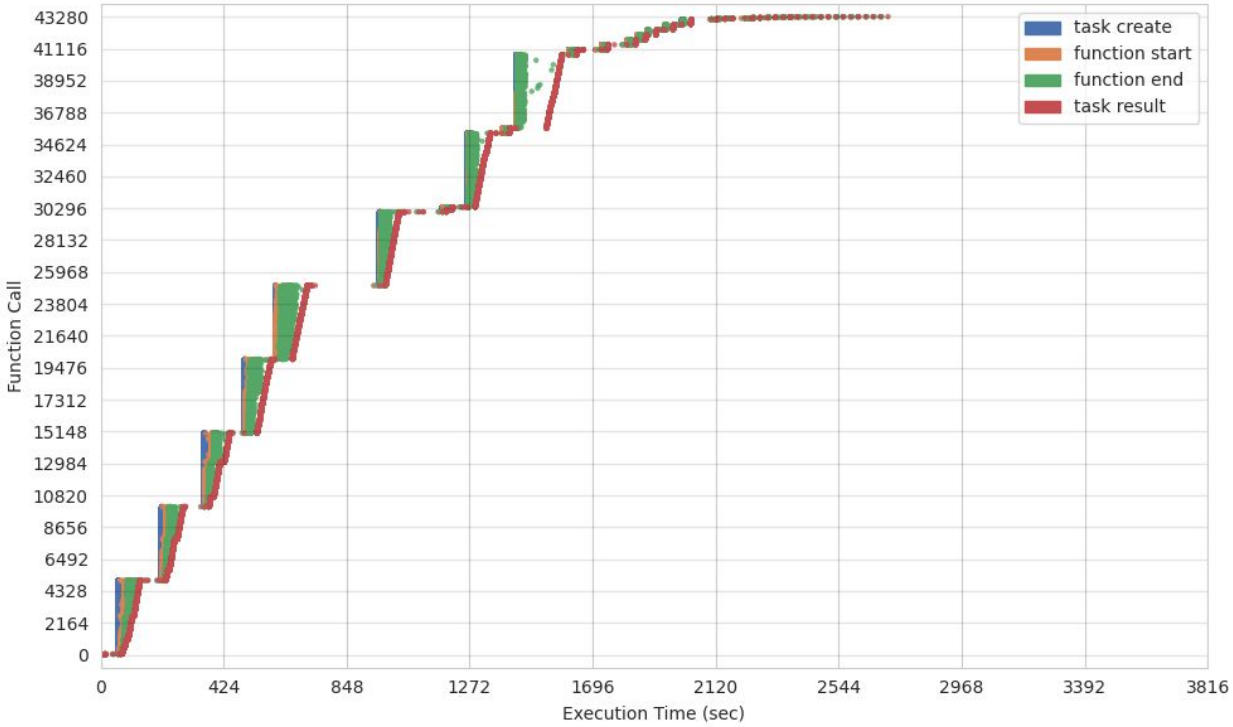
Overall design

- Bounded-Memory Serverless Array Processing



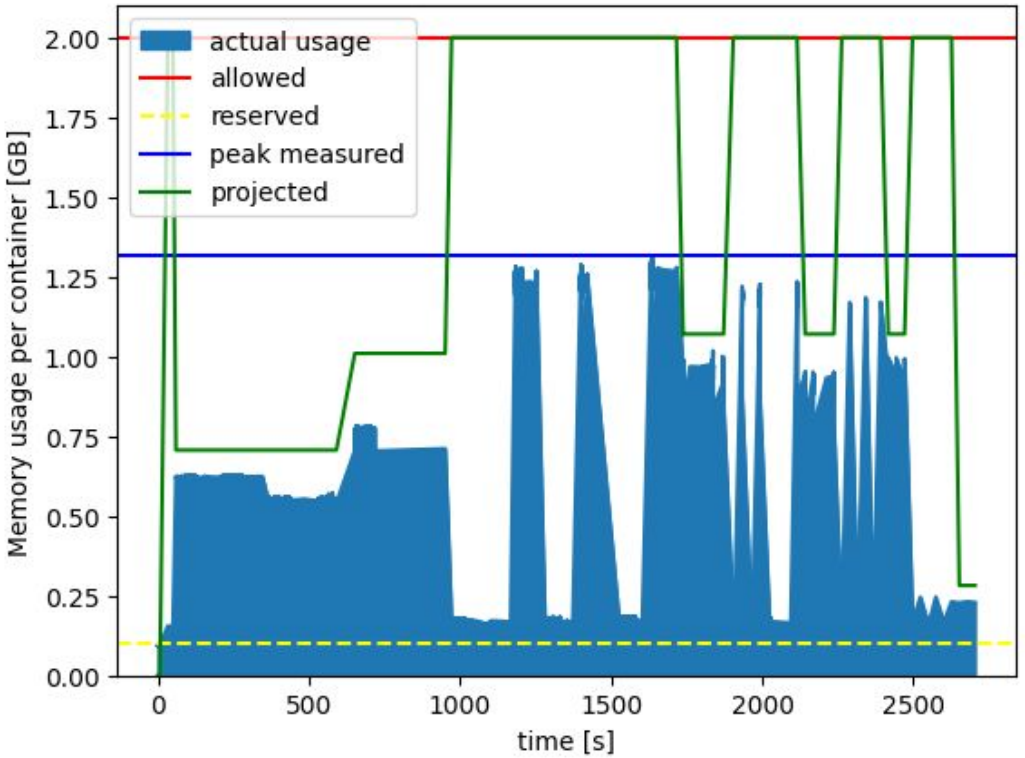
Initial results

- Tested on “quadratic means” problem
- Scales up to 1000 containers



Initial results

- Memory usage controlled
- Overall slower than dask + Coiled on same problem
- Room to optimize through task fusion!
- (Details in xarray blog post))



Xarray Integration

- Xarray has been generalized to wrap any chunked array type
- Install cubed & cubed-xarray
- Then specify the allowed memory
  - (And the location for intermediate Zarr stores)

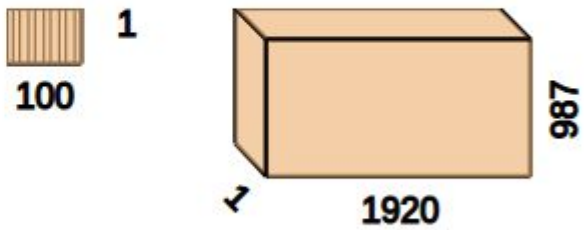
```
from cubed import Spec

spec = Spec(work_dir='tmp', allowed_mem='1GB')
```

Xarray Integration

- Now you can directly open from disk as `cubed.Array` objects

```
ds = open_dataset(
    'data.zarr',
    chunked_array_type='cubed',
    from_array_kwargs={'spec': spec})
    chunks={},
)
```

	Array	Chunk
Bytes	1.41 GiB	144.58 MiB
Shape	(100, 1, 987, 1920)	(10, 1, 987, 1920)
Count	4 arrays in Plan	10 Chunks
Type	float64	np.ndarray



Xarray Integration

- Now just .compute, with your chosen serverless Executor!

```
from cubed.runtime.executors.lithops import LithopsDagExecutor


ds.compute(executor=LithopsDagExecutor())
```

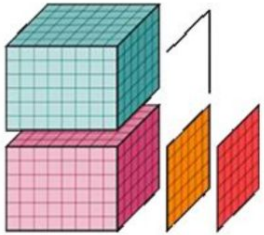
Xarray Integration

- Xarray now wraps Cubed OR Dask OR [new things??]

Tabular data:

Executes via

Array data:



xarray

Executes via

Cubed

??

Vision for Science at Scale (Tom's  list )

- Expressive ✅
- Horizontal scaling ✅
- No Cluster ✅
- Predictable RAM usage ✅
- Retry failures ✅
- Resumable ✅
- Fully open ✅

Disadvantages

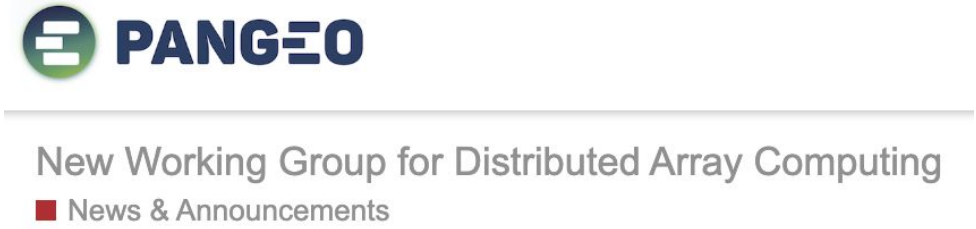
- I/O to Zarr is slow compared to ideal dask case of staying in RAM
- Serverless more expensive per CPU-hour
- Only array operations

Next steps

- We want your use cases to test on!
- Optimizations
- Other array types (JAX?)
- Other storage layers (Google-TensorStore?)
- Zarr v3+ new features

Read the blog post! xarray.dev/blog/cubed-xarray

- Join the discussion!



Thanks to Tom White for writing Cubed!